APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: HETEROGENEOUS NETWORK FILE ACCESS
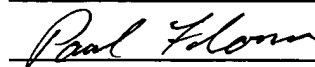
APPLICANT: KEVIN E. KALAJAN

HETEROGENEOUS NETWORK FILE ACCESS

BACKGROUND

The present invention relates generally to electronic communications.

5   During communication sessions in data networks, clients typically request information provided by servers. Increasingly, clients seek information located on different servers, coupled to networks having different data encoding and file transport protocols. For example, a user in an

10   organization using one computer (e.g., a windows-based machine), might seek to access data on another windows-based machine coupled to a local area network (LAN) running Novell NetWare, as well as data on a Unix-based machine, located on a LAN running Unix/NFS (Network File System), as well as

15   data residing on an Apple Macintosh machine, located on a LAN running AppleTalk.

One method for allowing such access requires development of a software application which both connects the two machines (the local client and the destination

20   server) as well performs all the translations in both data encoding and file transport at both ends of the communication link. Such a program, installed and running at both ends of the communication link, can translate the client's network protocols into the proper file access

25   protocols for the server's network, and then retrieve the desired file, while translating the file's data from its

original format into a format usable by the client. Each such connection between different types of client and server environments typically requires separate network and data translation schemes. In addition, such software

5 applications must typically reside at the network layer 3, and be installed into the kernel of the operating system of both the client and the server. Therefore, for a number of clients to communicate with a given server, such translation/communication applications typically requires

10 configuration and installation into each client's operating system.

"Tunneling" typically allows redirection of network drives, where a file received in one network protocol (e.g., NFS, Windows Networking (SMB) is typically encrypted within

15 another layer of redirection (the "tunnel"). Generally, tunneling is also implemented at the operating system kernel and is platform specific.

Web browsers can be used to initiate file transfers (e.g., with an "ftp://<url>" nomenclature), but this only

20 allows file transfers from a machine residing on the Internet.

SUMMARY

In general, in one aspect, the invention features a method for a client to access data files residing on a first

25 data server through a network, the method including coupling a heterogenous proxy server to the first data server through

2

a first local network protocol, receiving at the heterogeneous proxy server a data file from the first data server by employing the first network protocol, translating the data file into a format compatible with transmission

5   through the network, and transmitting the translated data file to the client across the network.

Embodiments of the invention may include one or more of the following features. A request can be sent from the client to the heterogeneous proxy server that the data file

10   be received from the first data server and then sent to the client. The heterogenous proxy server can be coupled to a second data server through a second local network protocol, the first and second local network protocols being different, and the heterogeneous proxy server can

15   selectively receive a data file from at least one of the first or the second data servers by employing the respective first or second local network protocols, can translate the data file into a format compatible with transmission through the network, and transmit the translated data file to the

20   client across the network. The network can employ Transmission Control Protocol (TCP). The format compatible with transmission through the network can be HyperText Transport Protocol (HTTP). The format compatible with transmission through the network can be a Multipurpose

25   Internet Mail Extension (MIME) of HTTP. The first and second local network protocols can each comprise one of the

3

following: Windows Networking (SMB), File Transport Protocol (FTP), Network File System (NFS), IPX/NCP (Novell Core Protocol), Banyan VINES, DECNet, and AppleTalk. The client can employ an HTTP browser for connecting to the

5   heterogeneous proxy server. File management and downloading services can be accomplished using an HTML document (or web page) containing information from the heterogeneous proxy server regarding available data files on data servers and providing the client with appropriate selections, including

10   allowing the client to send a request for the data file to the heterogeneous proxy server. Or the client can download an applet executable by the HTTP browser, the applet being configured to receive information from the heterogeneous proxy server regarding available data files on the data

15   servers, and to send a request for the data file to the heterogeneous proxy server. The applet, upon receiving the data file, can initiate an appropriate application for using the data file. The data file can be compressed at the heterogeneous proxy server before the data file is

20   transmitted to the client. The data file can be e-mailed from the heterogeneous proxy server to an e-mail recipient, without transmitting the data file to the client. The heterogeneous proxy server can search for the data file at one or more data servers. The client can be authenticated

25   before connecting the client to the heterogeneous proxy server.

4

In general, in another aspect, the invention features a method for a client to access data files residing on at least a first and a second data server through a network, wherein the network employs Transport Control Protocol

5   (TCP), the method including coupling a heterogenous proxy server to the first data server through a first local network protocol, and to the second data server through a second local network protocol, the first and second local network protocols being different, sending a request from

10   the client to the heterogeneous proxy server that the data file be received from the first or second data servers and then sent to the client, wherein the client employs an HTTP browser for connecting to the heterogeneous proxy server, selectively receiving at the heterogeneous proxy server a

15   data file from at least one of the first or the second data servers by employing the respective first or second local network protocols, translating the data file into a format compatible with transmission through the network, comprising HyperText Transport Protocol (HTTP), and transmitting the

20   translated data file to the client across the network.

In general, in another aspect, the invention features a storage device tangibly storing a control program, the control program, when coupled to a control device, operating the control device to allow a client to access data files

25   residing on a first data server through a network, the control program being configured to operate the control

5

device to perform the functions of coupling a heterogenous

proxy server to the first data server through a first local

network protocol, selectively receiving at the heterogeneous

proxy server a data file from the first data server by

5   employing the respective first local network protocols,

translating the data file into a format compatible with

transmission through the network, and transmitting the

translated data file to the client across the network.

Advantages of the invention may include one or more of

10   the following. A number of clients can connect to a

heterogeneous proxy server and gain access to files located

on a number of different network platforms. A client can

use a common web browser to connect to the heterogeneous

proxy server without special software in order to gain such

15   access. Moreover, heterogeneous clients using different

computer platforms (such as PCS, Unix machines or

Macintoshes) can access data on any number of different

servers, if the heterogeneous proxy server has an

appropriate protocol interpreter. No special translation

20   software is required at either the client or the target

server having a desired file -- all translation occurs at

the heterogeneous proxy server. Each user can seamlessly

and transparently retrieve a file, regardless of its native

network format or location, remotely retrieve it across the

25   Internet, and have their local web browser automatically

detect the type of file retrieved, load an appropriate

6

application, and then display the file to the user. Upon

completing operations with the file, the user can close and

save it, and the web browser can be configured to send the

file back to a destination data server (which can be

5   different from the original source data server), and the

file, before further transmission and storage at the

destination, can be retranslated by the heterogeneous proxy

server into the network format native to the destination.

Calculation and time intensive operations upon files

10   (such as compression and searching) can be conducted by the

heterogeneous proxy server, leaving relatively lower

bandwidth operations (such as list and file transfers) to

the connection between client and heterogeneous proxy

server. Network traffic between client and heterogeneous

15   proxy server can be automatically encrypted by SSL using

HTTP/S, incorporated in newer web browser applications, and

compressed by any compression schemes easily decoded via a

browser or associated helper applications. An

administrative application can be coupled or embodied in a

20   heterogeneous proxy application in order to capture, record,

and tabulate operations such as user accesses and server and

file accesses. The administrative application can thereby

keep a detailed logging and accounting of this usage data

for security monitoring and capacity planning. Furthermore,

25   a detailed logging of all Internet-based remote accesses can

simplify such data collection, as opposed to recording and

7

retrieving such data at each separate network data server being accessed.

These and other features and advantages of the present invention will become more apparent from the following description, drawings, and claims.

5

<div align="center">DRAWINGS</div>

Figure 1 is a schematic diagram of a network providing a heterogeneous proxy server and application.

Figure 2 is a functional block diagram of a

10   heterogeneous proxy application.

Figures 3a through 3h are flow charts illustrating the operation of a heterogeneous proxy application.

Figure 4 is a heterogeneous proxy application stored on a machine-readable device.

15

<div align="center">DESCRIPTION</div>

As shown in Figure 1, a client 10 connects across a network 12 to a heterogenous proxy server system 14, initially through a public server 16. In a preferred embodiment, network 12 is an IP-compliant network (e.g., the

20   Internet), and client 10 connects to public server 16 via HTTP using a "web" browser 20. However, the methods and apparatus described below can be used within other networks 12, using other network protocols and communication applications.

25   An access applet 22 can be downloaded from public server 16 and executed at client 10. The applet can be an

ActiveX or Java-based access applet 22 executed by browser
20. Access applet 22, in conjunction with public server 16,
can perform optional authentication procedures upon client
10, which, if successful, connect client 10 to secure server
24. Alternatively, instead of a downloaded access applet, a
user interface for performing functions with heterogeneous
proxy server system 14 can be provided by other front end
applications at the heterogeneous proxy server system 14.
For example, an HTML web page sent by either public server
16 or private server 24 to client 10 can allow for user
selection of any or all of the functions described below,
without downloading any higher level software.

Heterogeneous proxy application 26, running on secure
server 24, can connect client 10 with data files residing on
one or more data servers 28a, 28b, and 28c coupled to one or
more different networks 30a, 30b, and 30c. For example,
first data server 28a can be coupled to secure server 24 via
a Windows NT network 30a, while second data server 28b can
be coupled to secure server 24 via a Novell NetWare
network 30b, and while third data server 28c can be coupled
to secure server 24 via a Unix/NFS network 30c.

Heterogeneous proxy application 26 uses one or more
protocol interpreters 32a, 32b, and 32c to communicate with
respective data servers 28a, 28b, and 28c on respective
networks 30a, 30b, and 30c. Protocol interpreters 32a, 32b,
and 32c can reside within heterogeneous proxy

9

application 26, or be separate modules accessible by heterogeneous proxy application. These protocol interpreters 32 may be installed into the kernel of the secure server 24 (or more generally any server machine which

5   is part of heterogeneous proxy server 14). Protocol interpreters 32 provide networked access to files. In some cases protocol interpreters 32 can implement high-level file transfer protocols (e.g. FTP) and in other cases implement kernel-based drive/directory redirection protocols (e.g. NFS

10   or Novell/IPX/NCP).

As shown in Figure 2, heterogeneous proxy application 26 includes a download file module 34, an upload file module 36, a compress file module 38, a search file module 40, a mail file module 42, and a create directory

15   module 44. Download file module 34 enables client 10 to select and retrieve a file from a selected data server 28 on a network 30, through network 12, to the client 10. Upload file module 36 allows a reverse operation: client 10 can transfer a file from itself through network 12 to a selected

20   data server 28 on a respective network 30.

Compress file module 38 allows heterogeneous proxy server 14 to compress a selected file locally before further operations (such as downloading or e-mailing). Search file module 40 allows client 10 to search directories on one or

25   more of the supported, attached data servers 28 in order to find and select a file. Mail file module 42 allows a client

10

to e-mail a selected file directly from heterogeneous proxy server system 14, without first downloading the file to client 10 and then e-mailing it. Create directory module 44 allows a client 10 to create a file directory on any of the supported, attached data servers 28 in order to store one or more files.

As shown in Figures 3a through 3h, a heterogeneous proxy server method 100 typically begins with client 10 connecting to public server 16 via web browser 20 (step 102). Client 10 can access a user interface web page, download access applet 22 at this time, have access applet 22 already preinstalled, or can download access applet 22 after being properly authenticated as a legitimate client (step 104). For example, traveling employees could simply connect to their company's public web page, log onto a file access web page, or download access applet 22, and begin an authentication and file retrieval session, without requiring anything but a readily available web browser, and without making any changes to the operating system of client 10.

Once client 10 is properly authenticated, it is connected to secure server 24 (step 106). Secure server 24 can then initiate additional access control methods (step 108). One example of such methods is given in co-pending application serial number 08/928,360, filed September 12, 1997, entitled "Remote Access-Controlled Communication", incorporated herein by reference. Such methods can provide

11

further levels of secure access for external clients 10 to sensitive data residing on secure server 24 and data servers 28.

Once client 10 has the requisite access, secure server 24 builds and transmits to client 10 a list of available data servers 28 coupled to secure server 24 (step 110). The list can be compiled from, e.g., a local host file, a DNS server for TCP/IP data server hosts, from a Windows Name Server for data servers 28 coupled through a Windows Networking protocol, from a Novell NDS or Novell Bindery for data servers coupled through NetWare, or through comparable processes or data structures for DECNET or Banyan VINES-based servers. This list can be built and stored in advance, rather than be constructed on the fly, although pre-built lists can yield errors when, e.g., a data server is no longer connected due to a network fault. Access applet 22 can present to client 10 the built list of available data servers 28, preferably in a convenient form, such as a typical graphical interface file manager window.

One method for building a list of available data servers 28 begins by querying the local public server 16 and secure server 24 (e.g., primary domain controller and the local domain-name servers) to list all hosts that are in the current domain. Also, each connected Windows name server can be queried for all known Windows-based hosts along with the respective workgroups. Further, each Novell bindery or

12

NDS tree can be queried for all respective Novell hosts.

Other data servers coupled through other network protocols

can be queried similarly. Each of the results of these

queries are stored in a local host file accessible by

5   heterogeneous proxy application 26, which can sort the list

and remove any duplicates.

Once client 10 receives the list of available servers,

the user of client 10 can select a given server and provide

a given action (step 112). For example, client 10 can,

10   again through a GUI, allow a user to mouse click a given

icon for a data server 28, indicating that the user would

like to open that data server 28 and browse its directory

and file contents. The access applet 22 can optionally

require an additional password at this (or any other) point

15   to grant the user access to the selected resource. Another

option is that the user can query the secure server for a

list of available network resources.

Upon receipt of the client request, secure server 24

attempts to connect to the requested data server 28 (step

20   114). Connections are made through respective network

protocol interpreters 32. Thus, secure server 24 appears to

a respective data server 28 as a fully compatible network

peer. For instance, if data on a Unix/NFS data server 28b

is desired, second protocol interpreter 32b is employed to

25   control transfer of data to and from second data server 28b,

and cause secure server 24 to appear like a Unix/NFS client

13

to data server 28b. If the connection attempt fails for any reason, control returns to step 110 and secure server 24 attempts to build a new correct list of available data servers and an appropriate message is transferred across

5   network 12 (e.g., using HTTP) to client 10 that the selected resource is unavailable.

If the connection is successful, client 10 receives a listing of contents of the selected data server 28, e.g., in a graphical interface file manager type list of directories

10   and files (step 116 (this step is also referred to as "Menu" for recursive returns during method 100)). Access applet 22 can handle all such displays of information, and can capture user requests at client 10. The user of client 10, referring to the displayed contents of data server 28, can

15   make further requests and take further actions, e.g., through mouse clicks and keyboard data entry. Such requests are transferred over network 12 and accepted by heterogeneous proxy application 26 on secure server 24 (step 118).

20   If the accepted request is to open a displayed folder (step 120), control then returns Menu (step 116), and heterogeneous proxy application 26 descends one level down the folder hierarchy of the current data server 28, generates a new list of folders and files, and then sends

25   this new list over network 12 to client 10 for display to the user.

14

If the accepted request is to open a displayed file (step 122), control transfers to the download file module 34 (Go to A in Figure 3b, step 124). First, the requested file is retrieved from, e.g., data server 28a via the appropriate protocol interpreter 32a over respective network 30a (step 134). Second, the file type, e.g., text, gif, jpeg, or other is determined (step 136). Third, an appropriate Multipurpose Internet Mail Extension (MIME) type and subtype are calculated from the file type, in order to send the file to client 10 over network 12 as an e-mailed file with an appropriate content type. If the file is simply ASCII or text (including program source code), it can be sent as content/type = "text/plain". If the file is an image file (JPEG, GIF, etc.), then it can be sent as "image/<image type>". If the file type is "other", the file extension of the file is used to look up a corresponding MIME type for that extension, and the file can be sent with information specifying application/file extension information. That is, if no MIME type is found, then the file is sent as "application/<extension>". If there is no file extension, the file can be sent in a customizable mime type configuration as "application/octet-stream". The file is then sent via network 12 to browser 20 at client 10 (step 140), and operation returns to Menu (step 142).

As an example, client 10 can be a Macintosh computer employing a Macintosh-compatible web browser such as

Netscape Navigator, and can be requesting a Microsoft Word

document having a file extension ".doc" located on a Windows

NT data server 28. Heterogeneous proxy application 28

retrieves the Word file using a Windows NT protocol

5   interpreter 32 across the Windows NT network, determines

that it is a Word text document having a ".doc" file

extension, and transmits a proper MIME file via HTTP over

network 12 (the Internet) to client 10. Upon receipt of the

file, web browser 20 (or access applet 22) automatically

10   translates the MIME document into a proper Macintosh-

formatted Word file (since web browser 20 has already been

configured to handle Macintosh files), and opens the Word

file using a Macintosh-compliant Word Program (if

available). This depends upon transferring the file with

15   the proper MIME type and subtype so that browser 20 can open

it properly. All of these activities appear seamless to the

user of client 10.

If the client request is a file processing command (or

a "High IQ" command) (step 126), control transfers (Go to B

20   in Figure 3c, step 128) to one of several commands. If the

client request is a compress file command (step 144), then

control passes to the compress file module 38 (go to D in

Figure 3d, step 146) and a selected file is retrieved from

the selected data server 28 as set forth above (step 156).

25   Heterogeneous proxy application 26 then compresses the file

(step 158), and the file is then stored (e.g., locally at

the secure server 24, on the original data server 28, or elsewhere) with a new name (step 160). Storing the compressed file allows for retransmission of the compressed file should communication with either client 10 or an e-mail destination fail. Once compression is complete, control returns to Menu (step 162).

If the client request is a mail file command (step 148), then control passes to the mail file module 42 (Go to E in Figure 3e, step 150) and a selected file is retrieved from the selected data server 28 as set forth above (step 164). Heterogeneous proxy application 26 then prompts client 10 for a destination to e-mail the file (step 166), and determines the file type (step 168). Depending upon file type, the file is sent by invoking a Mailer routine to send the file to a requested destination with the proper file type (step 170). For example, if the file is simple ASCII, then it can be sent as an e-mail text file in the main body of the e-mail message. If it is binary, then the file can be encoded using Base64, made into an e-mail attachment, the user can be queried for proper addressees (including, e.g., any cc's, or bcc's, reply to, and other header fields), and then the properly addressed file can be sent. Once the file has been sent, control returns to Menu (step 172). Compression and mailing of a selected file can optionally be combined into a single command, such as

"compress and send", which would combine the two sets of procedures.

If the client request is a search for file command (step 152), then control passes to the search file module 40 (Go to F in Figure 3f, step 154) and client 10 is prompted for conventional search criteria such as text strings, wild cards, and/or boolean operations (step 174). Directories on a selected data server (or servers) are recursively searched to locate all matching files (step 176). This searching is conducted through commands from heterogeneous proxy application, translated and mediated by one or more of the appropriate protocol interpreters 32. Results of the search are compiled and then relayed via HTTP across network 12 for display to the user of client 10 (step 178). Again, control returns to Menu (step 180).

Referring again to Figure 3a, if the client request is to upload a file from client 10 to a selected data server 28 (step 130), then control passes to the upload file module 36 (Go to C in Figure 3g, step 132) and heterogeneous proxy application 26 receives the data file from client 10 via HTTP across network 10 (step 182). The received file is decoded based upon its encoding type (step 184). That is, Macintosh files are decoded from Binhex, Windows 95 files are decoded from Base64, and Unix files are decoded from Uuencode. Then, the appropriate protocol interpreter 32 is invoked to transfer the uploaded and decoded file to a

18

selected folder and filename on the respective selected data server 28 (step 186), where the file is then appropriately stored (step 188). At the conclusion of the file upload procedure, control again returns to Menu (step 190).

5   Referring again to Figure 3a, if the client request is to create a directory at a selected data server 28 (step 131), then control passes to the create directory module 44 (Go to G in Figure 3h, step 133) and heterogeneous proxy application 26 receives the server and directory information from client 10 via HTTP across network 10 (step 192). The received information is then used to create the appropriate directory at the selected data server 28 using the appropriate protocol interpreter 32 (step 194). At the conclusion of the create directory procedure, control again

15   returns to Menu (step 196).

Referring to Figure 4, computer program 210 (comprising executable instructions) can be placed upon any machine-readable device 200, such as a floppy disk, CD-ROM, removable hard drive, or other memory device, and can then

20   be loaded into secure server 24. Computer program 210 can include instructions which, when loaded into secure server 24 (or any other servers, including public server 16), provides the application software needed to generate an appropriate heterogeneous proxy application 26

25   to provide a heterogeneous proxy system 14.

Other embodiments are within the scope of the claims.
For example, the network 12 can be any electronic
communication medium. Client 10 and heterogeneous proxy
server system 14 can use any available electronic

5   communication protocol and interfaces. Data servers 28 can
be connected to heterogeneous proxy server system 14 through
any available network protocol or system, including wireless
radio and infrared networks. Any software applications at
the client and server can be implemented in software code

10   executed by one or more general purpose computers, in
firmware, or in special-purpose hardware. Heterogeneous
proxy application can include one or more of the specific
modules 34 through 44 described above, and other modules for
performing additional functions with data files received

15   from a data server 28.

What is claimed is: